SCGPSim: A fast SystemC simulator on GPUs

Mahesh Nanjundappa¹, Hiren D. Patel², Bijoy A. Jose¹ and Sandeep K. Shukla¹
¹FERMAT Lab, Virginia Polytechnic Institute and State University, ²University of Waterloo

ABSTRACT

SystemC promises an environment for faster hardware/software design-space exploration.

With systems becoming more complex, the simulation performance of the corresponding SystemC RTL design deteriorates significantly as current SystemC standard makes no effort to parallelize its simulation kernel to take advantage of multi-core platforms that are becoming the norm nowadays.

In this work, we present a translator that transforms synthesizable SystemC designs into parallelly executable programs targeting an NVIDIA graphics processing unit (GPU).

The translator retains the discrete-event semantics of the original designs by applying semantic preserving transformations. The resulting programs exploit the hardware parallelism for improved simulation efficiency.

Preliminary experiments show a simulation speed-up of approximately 30x-100x.

INTRODUCTION

➢ SystemC, a modeling and simulation language, has been used in making early trade-off analysis and design-space exploration.

➢ OSCI implementation of SystemC simulation kernel, being a Discrete Event Simulation Kernel makes no attempts to take advantage of recent parallel architectures.

➢ We present a source-to-source (S2S) translator that transforms SystemC designs into parallelly executable CUDA programs that run on NVIDIA Graphics Processing Units (GPUs).

➢ Preliminary experiments show impressive results by giving speed-ups from 30-100x on certain benchmarks.

SYSTEM TO CUDA TRANSLATION

➢ XMLization provides an intermediate representation of the SystemC design.

➢ Intermediate format is parsed to extract structural information of the design.

➢ Synthesizer/Translator uses semantic preserving transformations to automatically translate SystemC to CUDA.

PARALLEL SYSTEMC KERNEL

➢ Parallelism is extracted by mapping each runnable process to a thread belonging to different warp and executing multiple threads parallelly on GPU.

➢ To prevent race condition, Double Buffering mechanism is used.

➢ Barrier Synchronization is employed for synchronization of various threads.

CONCLUSIONS

➢ An automatic translation of synthesizable subset of SystemC designs into parallelly executable CUDA designs.

➢ Experimental results reveal an immense potential of GPU’s being used in EDA applications to increase the performance of the EDA tools.

➢ Future work on expanding the translator to be able to handle translations of designs beyond synthesizable subset of SystemC.

EXPERIMENTAL RESULTS

➢ Experiments conducted for the benchmarks listed in Table 1 and the corresponding time taken and speed-up obtained is notified in the table.

➢ SystemC GPU simulation was done on Intel Core2Duo processor running at 1.83GHz with 1.5GB of RAM.

➢ SystemC GPU simulation was done on NVIDIA’s TESLA D870 platform.

➢ Experiments were ran for 100,000 cycles and the results of both the methods were found to be same.

REFERENCES

4. NVIDIA Corp. NVIDIA CUDA. http://www.nvidia.com/cuda

Table 1: Experimental results for various examples

<table>
<thead>
<tr>
<th>Design</th>
<th>CPU (sec)</th>
<th>CPU+GPU (sec)</th>
<th>Speed Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipeline AES</td>
<td>120.1</td>
<td>3.916</td>
<td>30.66</td>
</tr>
<tr>
<td>FIR</td>
<td>51.9</td>
<td>1.37</td>
<td>37.68</td>
</tr>
<tr>
<td>10 Stage Buffer</td>
<td>28</td>
<td>0.277</td>
<td>101.08</td>
</tr>
<tr>
<td>Simple ALU</td>
<td>13</td>
<td>0.146</td>
<td>89.041</td>
</tr>
<tr>
<td>3 Stage Buffer</td>
<td>11</td>
<td>0.276</td>
<td>39.85</td>
</tr>
</tbody>
</table>

Graph 1: Time taken versus Number of cycles for pipeline AES example

Fig 1: A Parallel SystemC Simulation Kernel

Fig 2: Design Flow of SystemC to CUDA Translator

CONTACT

Mahesh Nanjundappa
FERMAT Lab, Virginia Tech
Email: knmahesh@vt.edu
Phone: 540-425-1151
Website: http://fermatlab.vt.edu/